A new taxonomy for reconfigurable prefix adders

Stevo D. Bailey
University of Virginia
Charlottesville, VA, USA

Mircea R. Stan
ECE Dept., University of Virginia
Charlottesville, VA, USA

***Abstract*—While previous taxonomies for prefix adders have focused on the design space of such adders with *fixed* topologies (in terms of fanout, radix, logic depth, wiring tracks), our work considers the design space of *reconfigurable* prefix adders (with applications in fault-tolerant adder design) by introducing several new degrees of freedom in the design space. Fault tolerance in general requires redundancy, so we start with a redundant structure which is a superset of the entire family of prefix adders with fanout-of-2 and then prune the redundant structure accordingly for the defect-free initial state or for when defects occur. In addition to the traditional prefix adders (Kogge-Stone, Han-Carlson, Brent-Kung) our taxonomy proposes several new variations that are equivalent in performance and complexity to the traditional structures, yet can mask different sets of faults.**

I. Introduction

Although parallel prefix adder topologies have been explored in the past [1,2], a need arises for a larger design space when aiming at reconfigurable adders capable of tolerating faults. If there is enough redundancy in the adder, a fault occurring at a specific node within the parallel prefix adder (PPA) can be bypassed by reconfiguring the actual structure of the adder [3]. Since many PPAs share similar nodes and designs, this reconfiguration can be achieved by moving a few edges (fine granularity) rather than duplicating entire adder structures as in triple-modular redundancy (TMR) schemes (coarse granularity). This paper proposes a unified design space for a set of such adder topologies which are useful for designing fault-tolerant, reconfigurable adders.

In a PPA the intermediate carries are computed in parallel with a prefix operation. By using two signals (generate and propagate) and a prefix operation, PPAs have a latency $O(\log_2 N)$ instead of $O(N)$, where N is the word length. The initial generate and propagate signals are *pre-computed* by

$$g_i = G_{i:i} = a_i \; AND \; b_i \qquad p_i = P_{i:i} = a_i \; OR \; b_i \qquad h_i = a_i \; XOR \; b_i \tag{1}$$

where a_i and b_i are the input bits. Additionally,

$$g_0 = G_{0:0} = C_{in} \qquad p_0 = P_{0:0} = 0. \tag{2}$$

The prefix operation, denoted by a dot operator, is defined as

$$(G_{j:m}, P_{j:m}) = (G_{j:k}, P_{j:k}) \bullet (G_{l:m}, P_{l:m}) \tag{3}$$

$$G_{j:m} = G_{j:k} \; OR \; (P_{j:k} \; AND \; G_{l:m}) \qquad P_{j:m} = P_{j:k} \; AND \; P_{l:m} \tag{4}$$

such that $j \geq (k, l) \geq m$. The dot operator is associative and idempotent, but not commutative [3, 4], and a prefix *tree* graph is used to calculate the carry bits based on (3) and (4)

$$c_i = G_{i:0} \tag{5}$$

while a *post-computation* is needed to produce the sum bits

$$s_i = c_i \; XOR \; h_i. \tag{6}$$

Several different PPAs graph trees have been previously proposed, with the most common being the Kogge-Stone, Han-Carlson, Ladner-Fischer, and Brent-Kung [1]. These adders have different logic depth, fanout, and wiring tracks (which can be viewed as three dimensions in a PPA design space [1]), in order to optimize area, delay, and power. A family of adders, somewhere between the Kogge-Stone and Ladner-Fischer, which has different fanout at each stage was described in [4]. Yet other dimensions in the design space are the valency or the radix of the prefix operation [5], and allowing irregularity [6]. Since increasing fanout causes significant delay increases in CMOS circuits, in this paper we focus only on PPAs with a fanout-of-2 (FO2) at each node (this is common practice in modern high performance adders [7]); also, to keep the design space exploration manageable, we consider only radix-2 structures. Fanouts and radices larger than 2 are topics for future work.

This paper develops a new taxonomy of parallel prefix adders designed for reconfiguration and use in fault-tolerant, nanoelectronic, CMOS integrated circuits. The taxonomy is described in the next section. A notation system for the new adders is proposed, followed by a description of several novel structures. Adders which are slightly irregular within the conventional design space but useful for reconfigurability are proposed in section III. Section IV contains the results, and section V concludes the paper.

II. Regular Taxonomy for Reconfigurable Adders

The new adder structures presented here come from a desire to trade the speed of a Kogge-Stone adder for the sparsity of a Brent-Kung adder while also being able to modulate around faults at prefix nodes, where sparsity is necessary for bypassing a faulty node without extra overhead.

The starting point for our design space is a *redundant* structure that is a combination of a full Kogge-Stone *forward*

graph with a full *reverse* graph as shown in Fig. 1. This graph incorporates *all the necessary nodes and edges* that constitute the Kogge-Stone, Brent-Kung, and all the FO2 radix-2 regular adders in between. Thus the idea of our work is that we can always start with the redundant full tree and then prune it until it becomes irredundant. When there are no faults, in general we will prune to a Kogge-Stone adder if we are interested in the highest performance, or to some other conventional structure (e.g. Brent-Kung) when we want lower power at the expense of a small increase in delay. Although it is interesting to be able to reconfigure such an adder to get different trade-offs between performance and energy, being able to reconfigure becomes really useful if we also consider the effect of defects/faults. In that case we can use the same initial full graph and prune it to an irredundant structure that does not include the faulty node/edge. To fully exploit this feature, though, it is necessary to consider other structures in addition to the traditional prefix adders. Each of the new adder structures only uses a subset of all these possible prefix nodes available, with the rest being bypassed. If a fault occurs at a specific node and is detected and localized, the adder can rearrange itself to bypass that node. The next steps are to determine which configurations are available and what notation can be used to classify them.

Fig. 1 shows the full graph with all nodes and edges turned on. A node is shown as a dot, and an edge is a line connecting two nodes. An active node is defined as one which performs the prefix operation, and is represented by a black dot. An inactive node (white dot) simply passes the other P and G signals to the output. The top row of black squares performs the pre-computation step (1). The post-computation (6) is not shown. From here on, we will refer to a "row" as a row of edges, not nodes. Since the graph in Fig. 1 is symmetrical with respect to the middle row, rows which span the same number of columns are referred to as *complements*. So the top and bottom rows are *complements*, as are the second and second to last rows, etc. The actual number of logic levels is equivalent to the number of nonempty rows, and for this paper, we define the normalized logic depth as in [1] such that the Kogge-Stone always has a logic depth of zero.

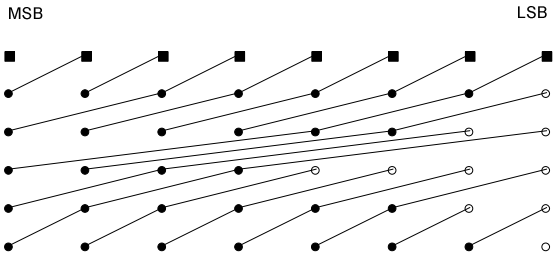

Figure 1. A *full tree* redundant graph

Fig. 2 shows the seven possible *regular* adders derived from the full topology seen in Fig. 1. A simple definition of a regular adder is one in which every row has a *periodicity* with period given by (8) and no wrap-around. The full tree has a period of 1 on every row. Fig. 2a is the Kogge-Stone. Two versions of a Han-Carlson adder are shown in Figs. 2b and 2c, and four variations of the Brent-Kung adder can be seen in Figs. 2d-g.

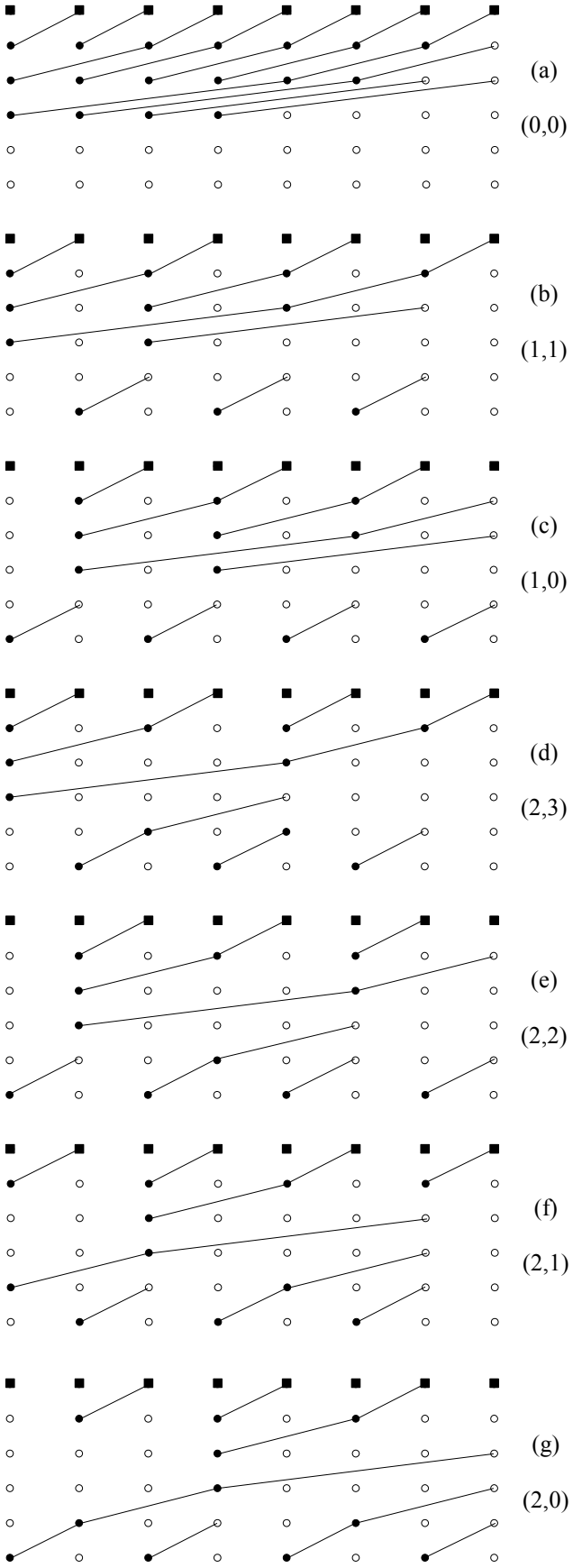

Figure 2. All seven regular irredundant graphs

Two differentiating characteristics for these adders are the *logic depth* and the *initial column* of the middle row. We use the convention from Harris [1] and denote the normalized logic depth, L, as varying from *0* to $log_2(N-1)$, where N is the word length. L is normalized, meaning that when L is zero, the actual number of logic levels is equal to the number of logic levels in the Kogge-Stone (log_2N), while when L is $log_2(N-1)$ the actual logic depth is the same as for a Brent-Kung adder ($2log_2(N-1)$). The *initial column of the middle row*, S, varies from *0* to 2^L-1. Fig. 2 has the values of (L,S) next to each graph. Although the (L,S) values uniquely define each structure, converting the notation into a valid adder structure is not trivial. The procedure to obtain the corresponding adder from a given (L,S) pair is described next.

When building an adder from a given (L,S) pair, the following formulas will be useful. First, let r be the row number, with $r = 0$ indicating the first row. Each row has a span, P. The period of a row is defined as how often the edges occur per node, and it is given by (8) where *min(...)* returns the minimum of the arguments. The starting column of a row, C_0, is defined as the column from which the first edge begins, with zero being the rightmost column and the LSB of the input. If $r < L$ then the row has a complementary row with starting column, C_{comp}, given by (10). The complementary row has the same span and period of the original row. The formulas are summarized below.

$$P = 2^r \tag{7}$$

$$T = min(2^L, 2^{r+1}) \tag{8}$$

$$C_0 = \begin{cases} (S + 2^r)\ modulo\ (2^{r+1}), & r < L \\ S, & r \geq L \end{cases} \tag{9}$$

$$C_{comp} = (T/2 + C_0)\ modulo\ T, \qquad r < L \tag{10}$$

To build a row, simply start at column C_0 and draw an edge spanning P nodes (right to left). Then every T nodes, draw another edge spanning P nodes unless the edge ends on a column greater than $N-1$ (since there are only N columns). Do the same for C_{comp}, if applicable. As an example, building a (1,1) 8-bit adder row-by-row is shown in Fig. 3. The regular adders presented here are irredundant, thus have a unique minimal number of edges given a logic depth and word length, meaning a unique minimum number of prefix operations, which minimizes the power consumption. Using the normalized logic depth, L, the minimum number of connections for a given L and N is found by

$$\sum_{i=0}^{log_2(N)-1} \frac{N - 2^i}{\min(2^i, 2^L)} \tag{11}$$

where *min(...)* is the minimum of the arguments. The number of regular structures is also a useful computation. It depends solely on N and can be found by

$$\sum_{i=0}^{log_2(N)-1} 2^i. \tag{12}$$

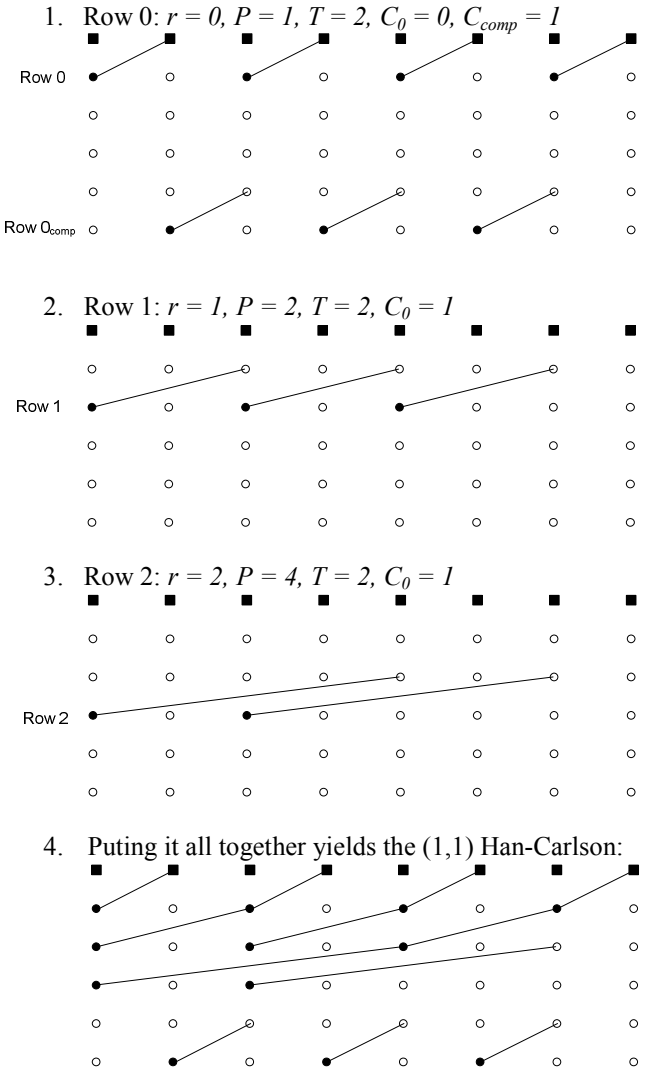

Figure 3. Steps for creating the (1,1) adder structure

III. IRREGULAR ADDERS

There are several other structures which can be derived from the full graph and provide more choices. These adders are termed irregular because they lack periodicity in every row. The first set of irregular adders comes from noticing that sometimes the *first edge* in a row with a complement may be switched to its complement row. For example, in (1,0), the rightmost edge in the last row can be moved to the first row without harming functionality. Additionally, in (2,0) and (2,1), the rightmost edge in the second to last row can move to the second row and still produce a functional adder. The change in (2,1) can be seen in Fig. 4a. The second set of irregular adders comes from partially converting a Kogge-Stone adder to a Han-Carlson adder. Every other column in the Han-Carlson is equivalent to that of the Kogge-Stone. To get a hybrid structure, only convert some portion of the adder to Han-Carlson, starting from the left as can be seen in Fig. 4b.

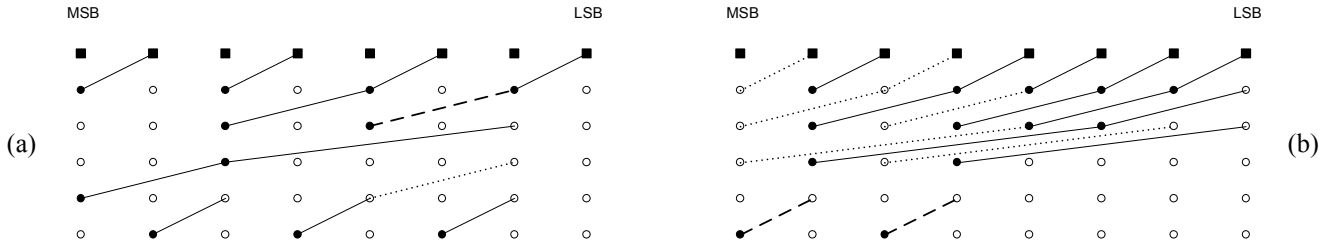

Figure 4. Ireggular adders: dashed edges replace the dotted edges: (a) modified (2,1) adder; (b) hybrid Kogge-Stone/Han-Carlson

IV. RESULTS

The starting point for reconfiguration depends on the application: we start with Kogge-Stone for high performance and with Brent-Kung for low power. When a fault occurs, we assume it happens in an active prefix node (a pessimisstic assumption), requiring a reconfiguration into a new structure. For simplicity, it is assumed that the adder will reconfigure into another regular adder. The Kogge-Stone will migrate to a Han-Carlson to maintain the highest performance, while the Brent-Kung will migrate to another Brent-Kung adder to retain low power. All possible single faults can be corrected for any initial adder. A conversion from a Kogge-Stone to a Han-Carlson has been documented previously in [8]. However, if another fault occurs at an active node, the adder presented in [8] would fail, having already reconfigured itself once. An adder using the topology of this paper seen in Fig. 1 could further reconfigure itself into one of the other regular adders or an irregular adder, such as those in Fig. 4. The adder chosen for the second reconfiguration would depend on the location of both faults and the desired performance-power trade-off. Fig. 5 shows two examples of complete adder reconfigurations: the solid path is for high performance and the dotted path is for low power. The gray arrows indicate the possible paths, which in general could be between any two adders. Next to each path is the faulty node, given by (*row*, *column*) where *row* contains the nodes at the *end* of the edges (in this case row refers to a row of nodes).

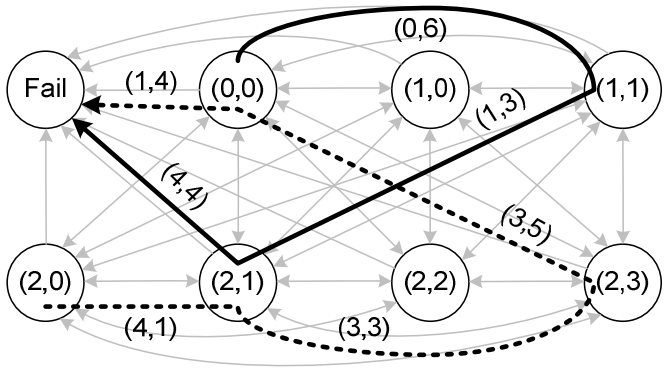

Figure 5. Possible adder reconfigurations: the solid path shows a high performance while the dashed path shows a low power reconfiguration

Table I shows the minimum and maximum sequential, correctable faults occurring on active nodes for a given word length and for two possible starting adders. It also gives the expected number of faults corrected by the adder for 8-bit words. A program was written in JAVA to simulate every possible combination of faults and how an adder might reconfigure around those faults, thus giving the data in Table I. The newly proposed adders can correct for more than two sequential faults on average, greatly improving upon existing fault-tolerant adders which can only handle one fault [8,9].

TABLE I. SEQUENTIAL FAULT ANALYSIS

N	**min # faults**		**max # faults**		**average # faults**	
initial adder	*(0,0)*	*(2,0)*	*(0,0)*	*(2,0)*	*(0,0)*	*(2,0)*
4	1	N/A	1	N/A	1	N/A
8	1	1	4	6	2.3	2.6
16	1	1	≥ 7	≥ 7		
32	1	1	≥ 8	≥ 8		

V. SUMMARY

A set of adders have been presented which can be used in a reconfigurable structure to bypass faults within the prefix nodes of a PPA. A design space encompassing the adders consists of two metrics, logic depth and starting column of the middle row, *L* and *S* respectively. For an 8-bit adder, this reconfigurable design can correct a maximum of 6 faults and an average value of 2.6 faults when starting with the (2,0) Brent-Kung adder.

REFERENCES

[1] D. Harris, "A taxonomy of parallel prefix networks," Proceedings of the 37th Asilomar Conferecnce on Signals, Systems, and Computers, pp. 2213-2217, 2003.

[2] M. M. Ziegler and M. R. Stan, "A unified design space for regular parallel prefix adders," DATE 2004.

[3] M. Guevara and C. Gregg, "Fault-tolerant, real-time reconfigurable prefix adder," unpublished.

[4] S. Knowles, "A family of adders," Proceedings of the 15th IEEE Symposium on Computer Arithmetic, pp. 277-281, June 2001.

[5] A. B.-Smith and C.-C. Lim, "Parallel prefix adder design," pp. 218-225, ARITH 2001.

[6] J. Liu, Y. Zhu, H. Zhu, J. Lillis, and C.-K. Cheng, "Optimum prefix adders in a comprehensive area, timing and power design space," ASPDAC 2007.

[7] J. Rabaey, A. Chandrakasan, and B. Nikolic, Digital integrated circuits, 2nd ed., Upper Saddle River: Pearson, 2003, p. 508.

[8] P. Ndai, S.-L. Lu, D. Somesekhar, and K. Roy, "Fine-grained redundancy in adders," pp. 317-321, ISQED 2007.

[9] S. Ghosh and K. Roy, "Novel low overhead post-silicon self-correction technique for parallel prefix adders using selective redundancy and adaptive clocking," IEEE Transactions on VLSI Systems, vol. 19, no. 8, pp. 1504-1507, 2011.